



CPEN 311: Digital Systems Design

Circuit Timing: Part 1: The Basics

Learning Objectives

1. Understand the RC model for gate delay and where it comes from
2. Be able to find the critical path, and hence the maximum clock speed, of a logic circuit (with or without flip-flops)
3. To understand set-up time, clk-q time, and hold time of a flip-flop
4. Understand setup and hold timing requirements of a design
5. Given a circuit, be able to calculate the maximum clock frequency
6. Given a circuit, be able to calculate the minimum hold time on the flip-flops

The Need for Speed

Speed (“Performance”) of your circuit is one of the top two optimization goals (power is the other)

Often, a circuit has strict timing constraints:

- Throughput constraints (eg. handle 10 GB/sec sustained)
- Latency constraints (total time required for answer)

Time perform an operation = **# cycles** * **cycle time**

- **# cycles** depends on FSM/control
- **Cycle time** is the focus now

This slide set: Basics of timing analysis

Coming later: Practical considerations



What does delay mean in hardware?

- “the time for electrons to get through a gate”

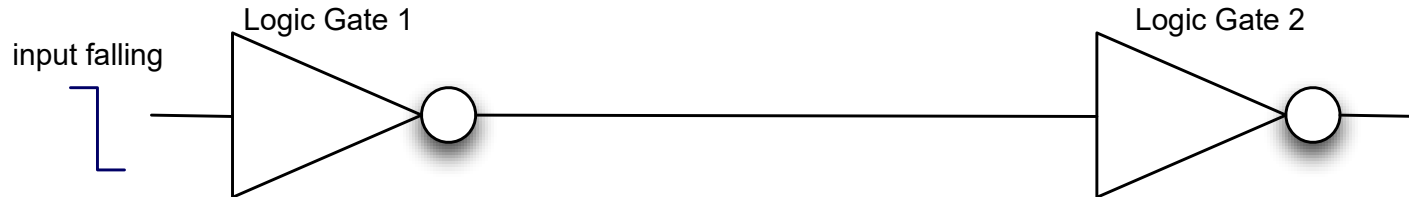
That is not quite right. Let's try again:

- “the time for electrons to fully charge the capacitance”

That's better ... but let's get a bit more precise.

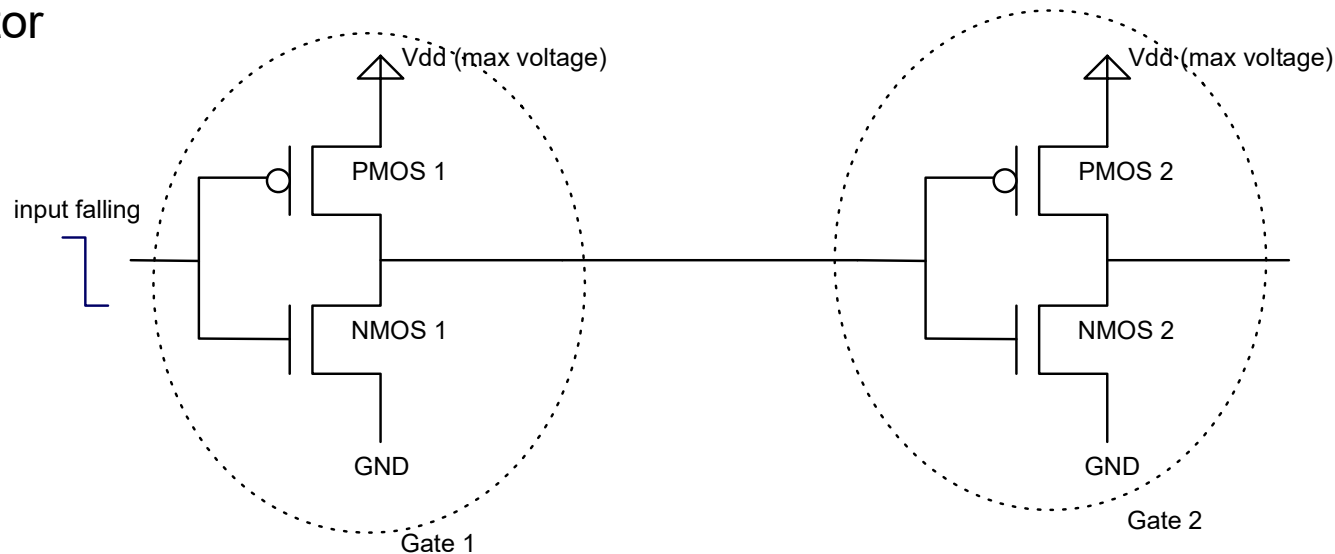
Fundamentals: Delay of a gate

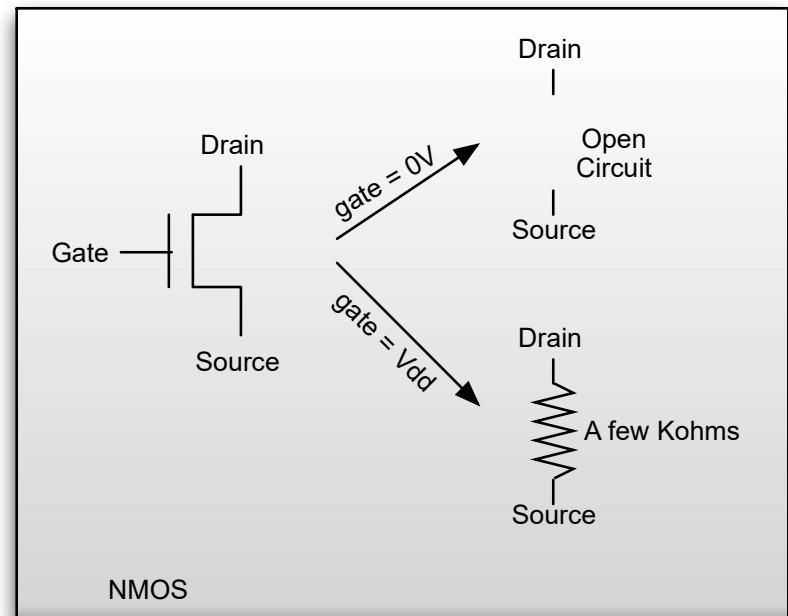
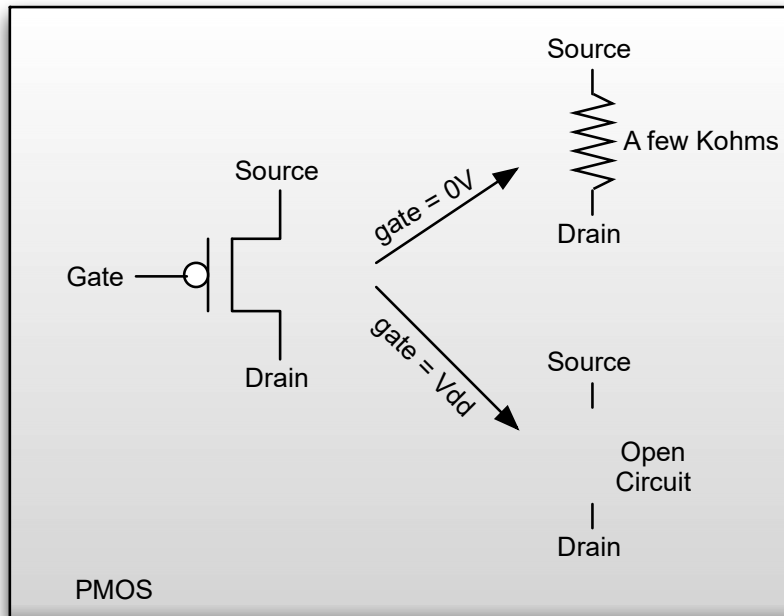
Consider Logic Gate 1 driving Logic Gate 2:



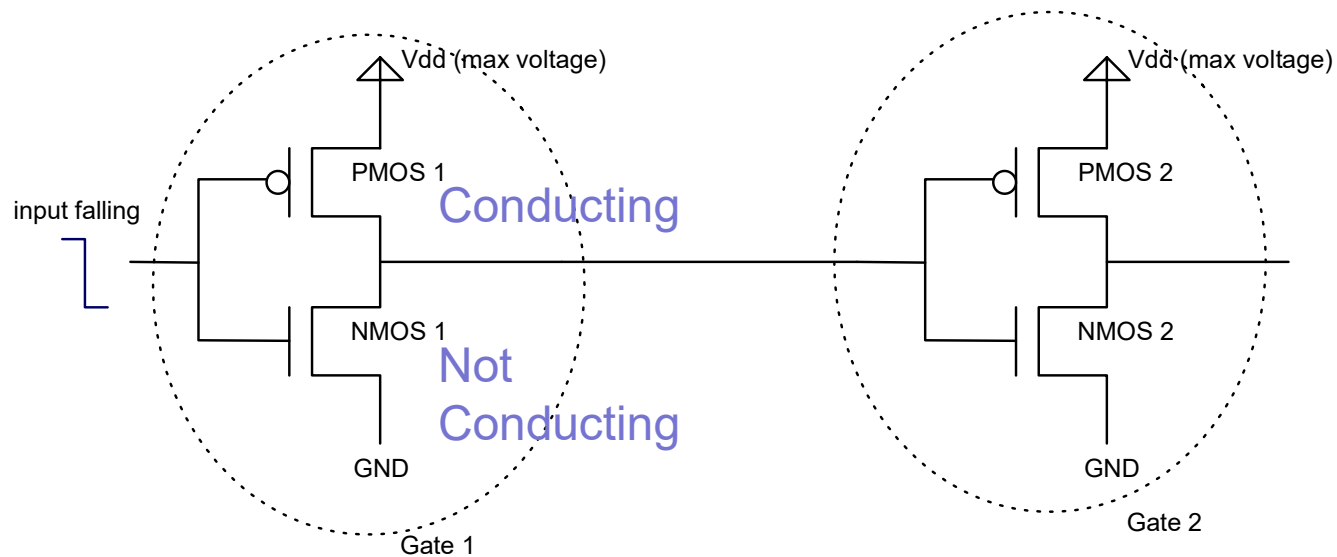
Electrically, what happens?

Recall from CPEN 211, and inverter consists of one PMOS and one NMOS transistor



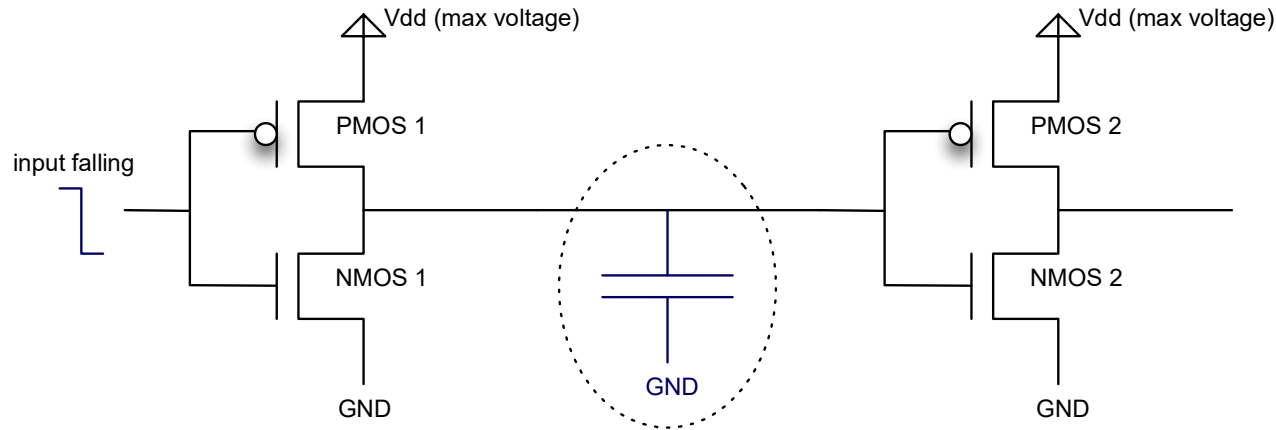


So what state are the transistors in **when input is 0?**

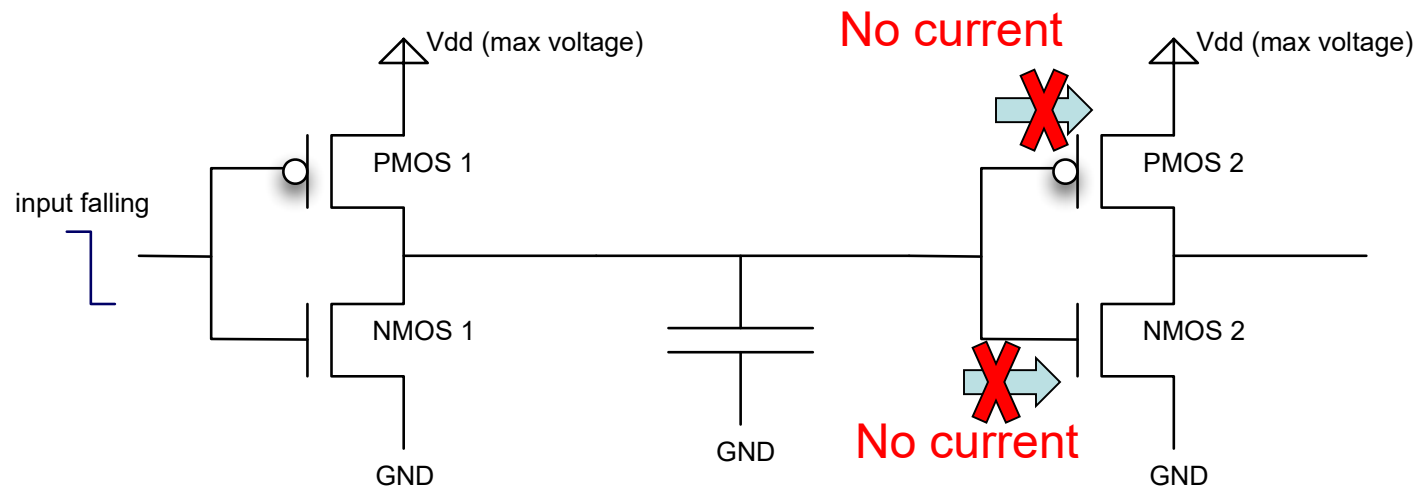


Two other facts you need to know:

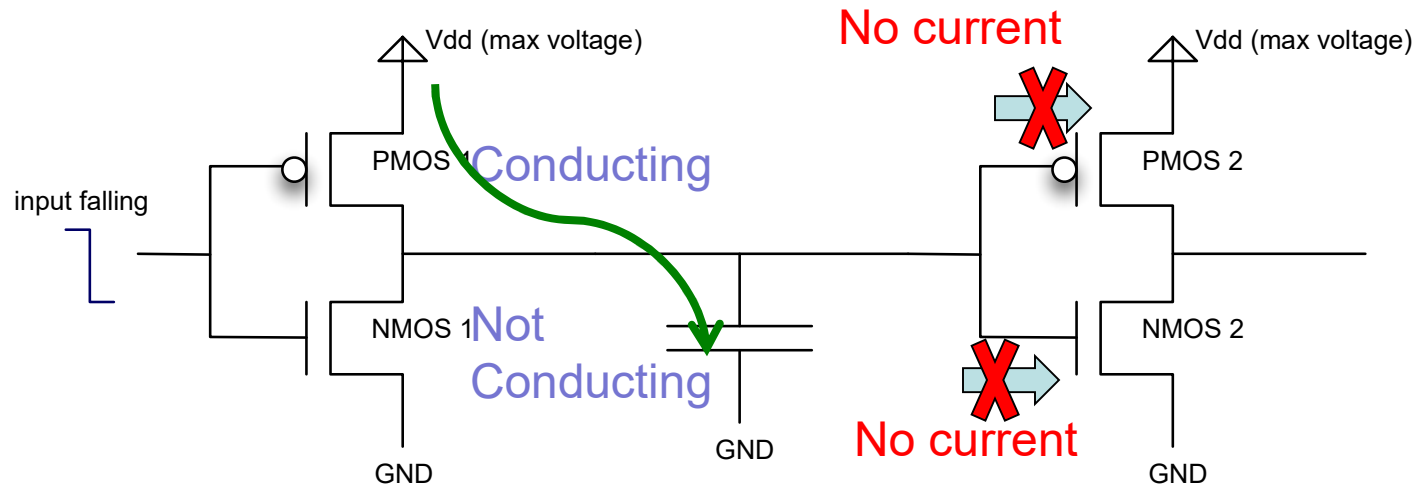
1. There is a “**parasitic capacitance**” attached to every wire and transistor
 - Due to the physical transistor structure



2. No current goes through the gate of a PMOS or NMOS Transistor



Given all that, what happens when input to first gate goes to 0?



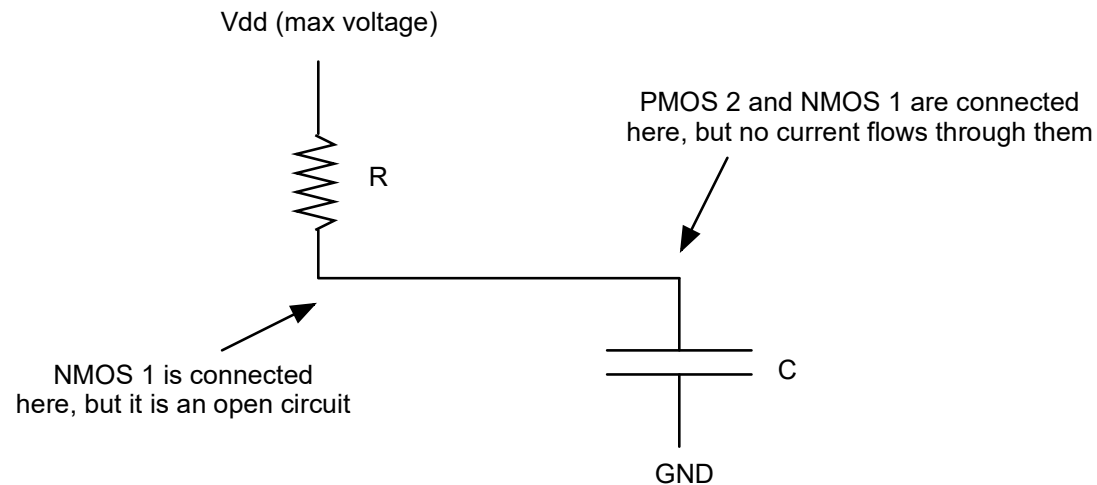
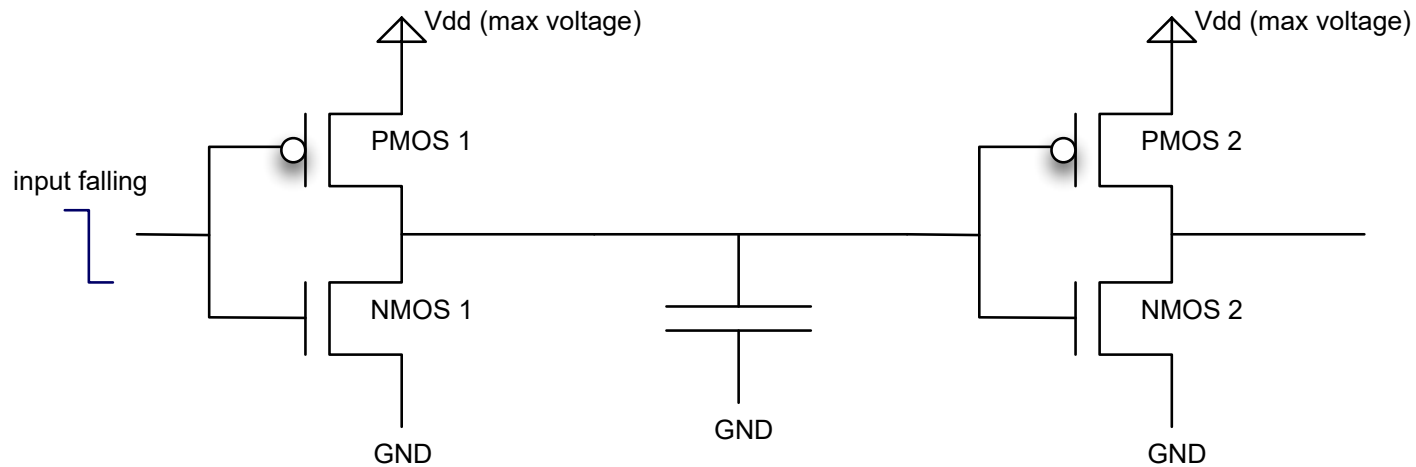
1. Initially the output of the first gate is 0.
2. When the input of the first gate falls, PMOS 1 becomes a resistance.
3. A current flows from Vdd, through PMOS 1, and charges the parasitic capacitance.
4. At some point, the parasitic capacitance reaches the **threshold voltage** of the second gate. At this point, the first gate is deemed to have “switched”, and the second gate starts switching.
 - “threshold voltage” is typically halfway between Vdd and GND

Observation: The “delay” of a logic gate 1 is the time for logic gate 1 to charge the capacitance on its output.

Note: this has nothing to do with electrons “passing through” the logic gate.
(That does not happen.)

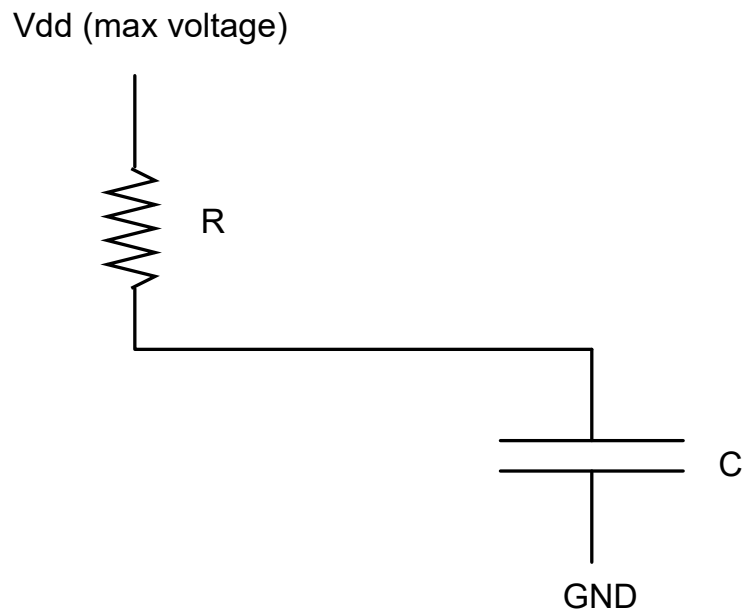
Transistor-Level Timing

We can go farther and create an “equivalent circuit”:



Transistor-Level Timing

How long does it take to charge the node?



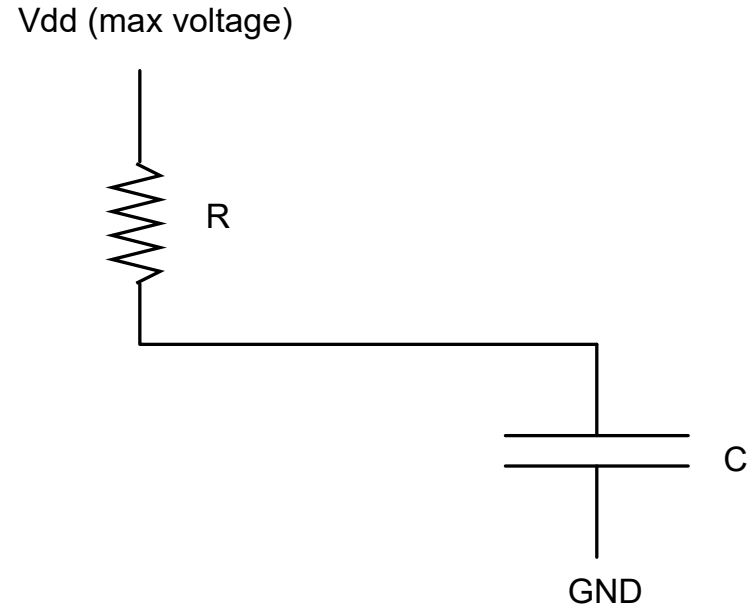
Initial condition output node = 0V, final condition is output node = $V_{dd}/2$

Works out to be proportional to $R \cdot C$

Transistor-Level Timing

Delay of 1 gate is proportional to $R \cdot C$.

Delay of n gates is proportional to $n \cdot R \cdot C$



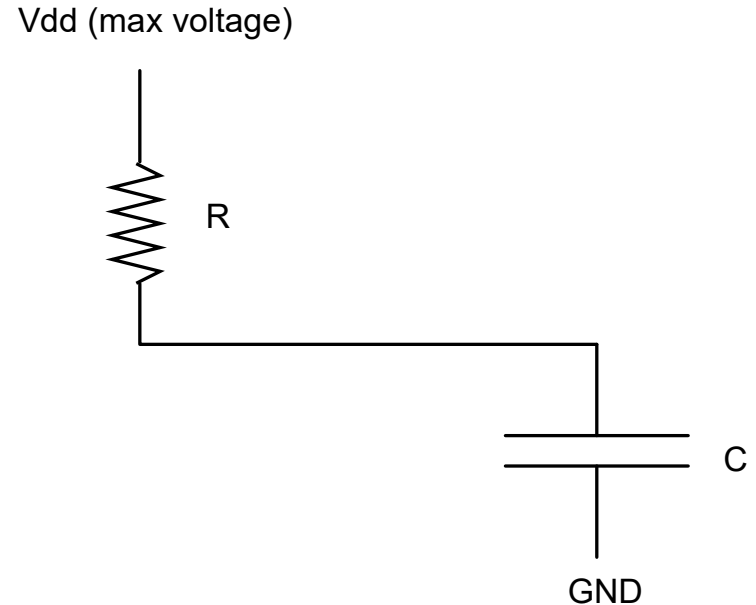
What does R depend on?

1. Size of transistors in the logic gate (bigger transistors = lower R)
 - The FPGA has fixed size transistors; you can not change them
2. Length of wire: longer wires have higher resistance
 - Placement step in Quartus Prime has some control over distance by locating source and destination nodes close together

Transistor-Level Timing

Delay of a gate is proportional to $R \cdot C$.

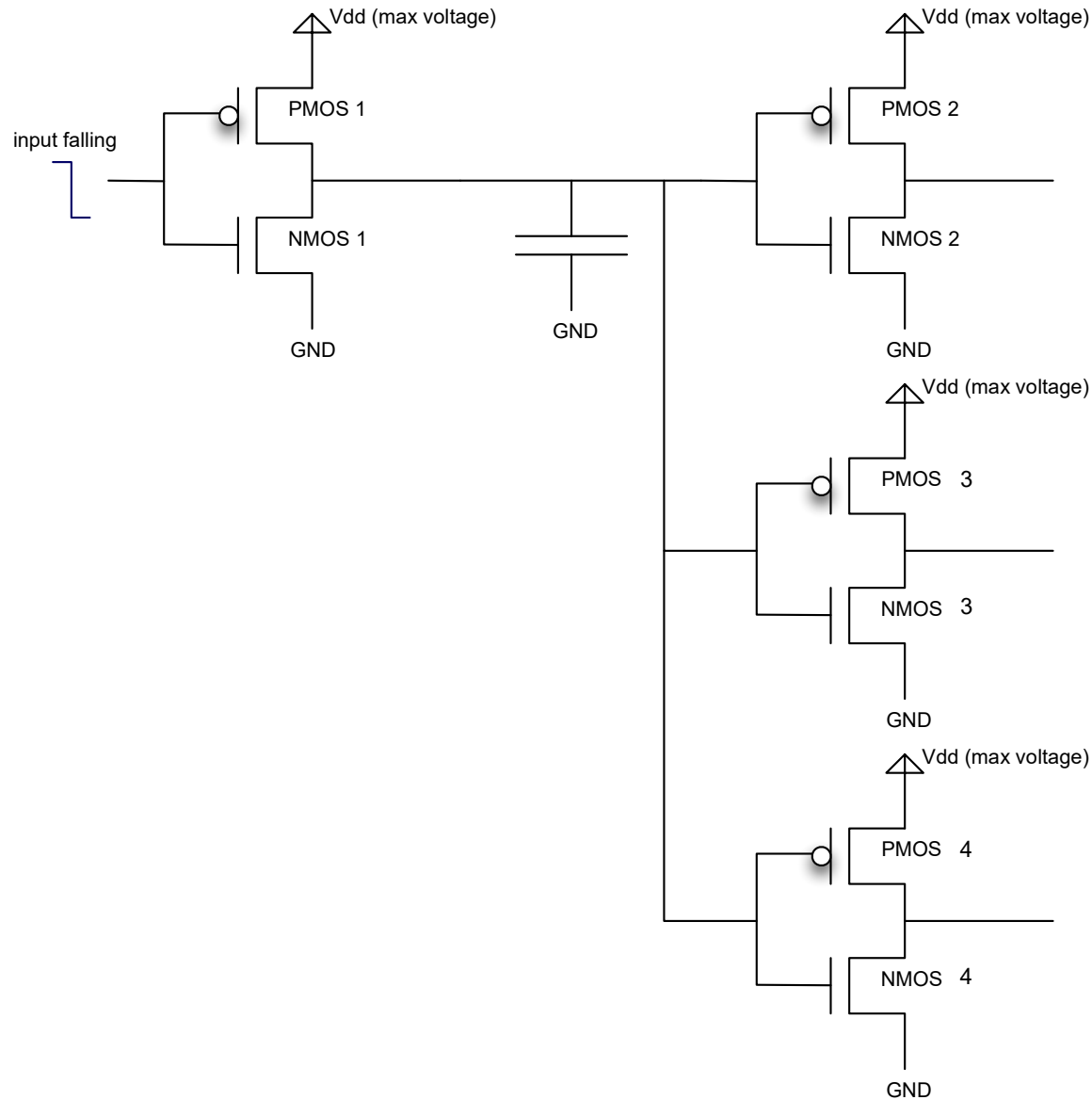
Delay of n gates is proportional to $n \cdot R \cdot C$



What does C depend on?

1. Size of transistors in the logic gate (bigger transistors = higher C)
 - The FPGA has fixed size transistors; you can not change them
2. Number of gates driven (fanout): see next slide
3. Length of wire: longer wires have higher capacitance
 - Another reason to place source and destination nodes close together

Fan-Out



The first gate drives 3 gates.

Each destination gate provides some parasitic capacitance. Since these caps are in parallel, you add them.

Total capacitance $\sim 3x$

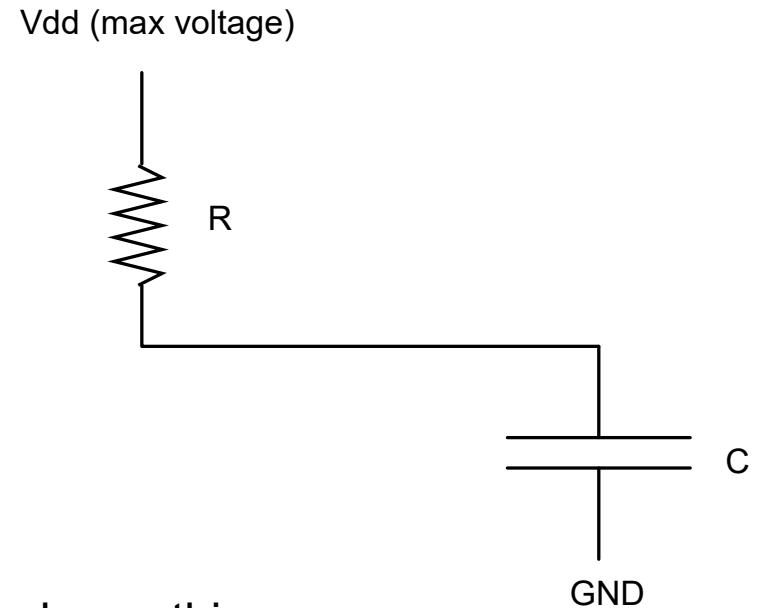
Transistor-Level Timing

Delay of 1 gate is proportional to $R \cdot C$.

Delay of n gates is proportional to $n \cdot R \cdot C$

What does n depend on?

Number of gates in the path. You have control over this.



Take-Away from the previous slides:

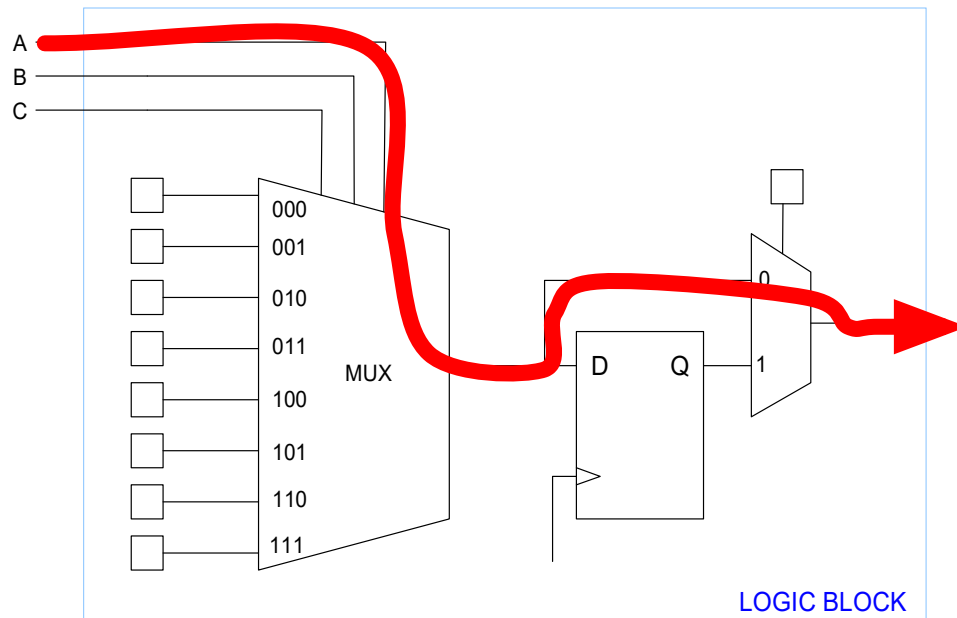
1. Delay of a logic gate depends on:
 - i. physical structure of gate
 - ii. length of the interconnect wire
 - iii. how many gates are driven.

2. To find path delay, add delays of all gates in a path

To learn a lot more about this, take ELEC 402 next year

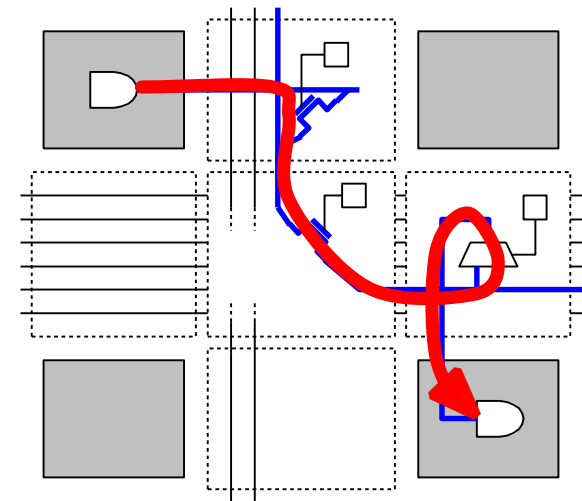
Where does Delay Come From in an FPGA?

Logic Delay:



On the order of 0.1 ns

Routing Delay:



On the order of 1 ns

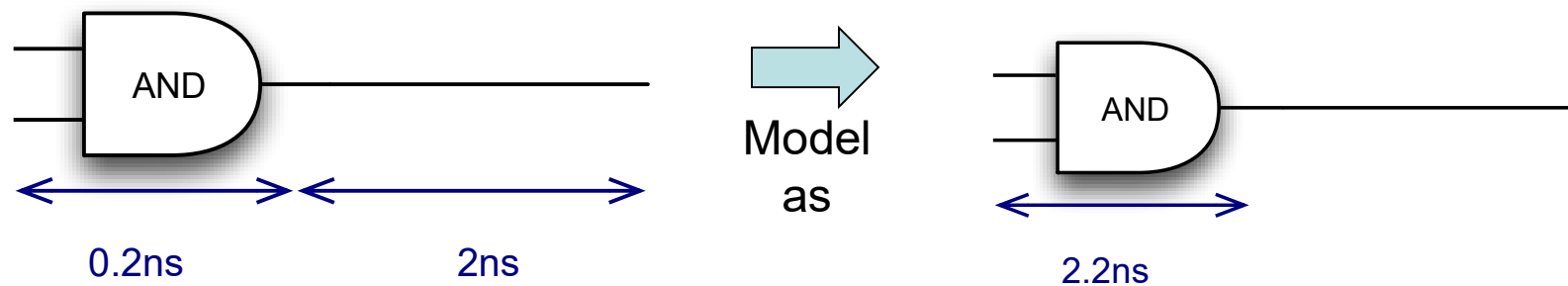
Delay Modeling

Given the transistor-level description of the FPGA device,
you *could* calculate all the Rs and Cs to calculate delay

- Often use a simulation tool (SPICE) which operates at transistor level
- Not feasible for large circuits
 - Abstraction: pre-calculate delays of gates

From the previous slide → routing delay dominates

- For simplicity
 - Routing delays and gate delays are sometimes lumped together
 - we may call these “gate delays”



Timing Concepts and Terminology

Combinational Timing Constraints

- Gate Propagation Delay (eg, 0.025ns = 25ps)
- Critical Path Delay (eg, 1.25ns)

Sequential (Flip-Flop) Timing Constraints

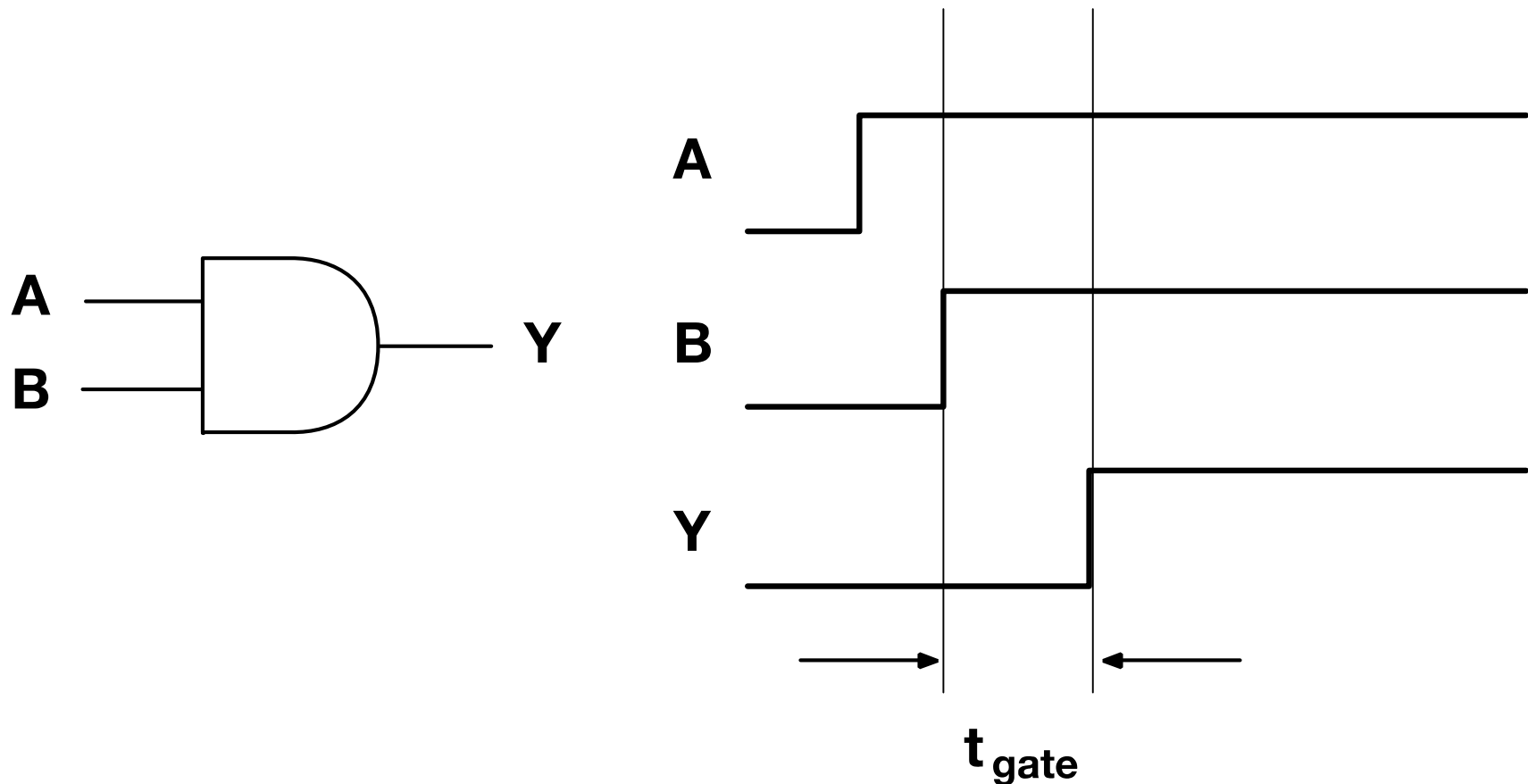
- Clock-to-Q Delay (eg, 0.02ns)
- Setup Time (eg, 0.6ns)
- Hold Time (eg, 0.05ns)

Circuit Speed

- Minimum Clock Period (eg, 1.25ns)
- Maximum Clock Frequency (eg, 800 MHz)

Gate Delay (Propagation Delay)

Gate Delay: Time it takes for combinational gate output to change after its last input changes



(it's actually a bit more complex: gate delay can be specified separately for each input, and delay may differ when the output transitions 0-to-1 or 1-to-0, etc etc)

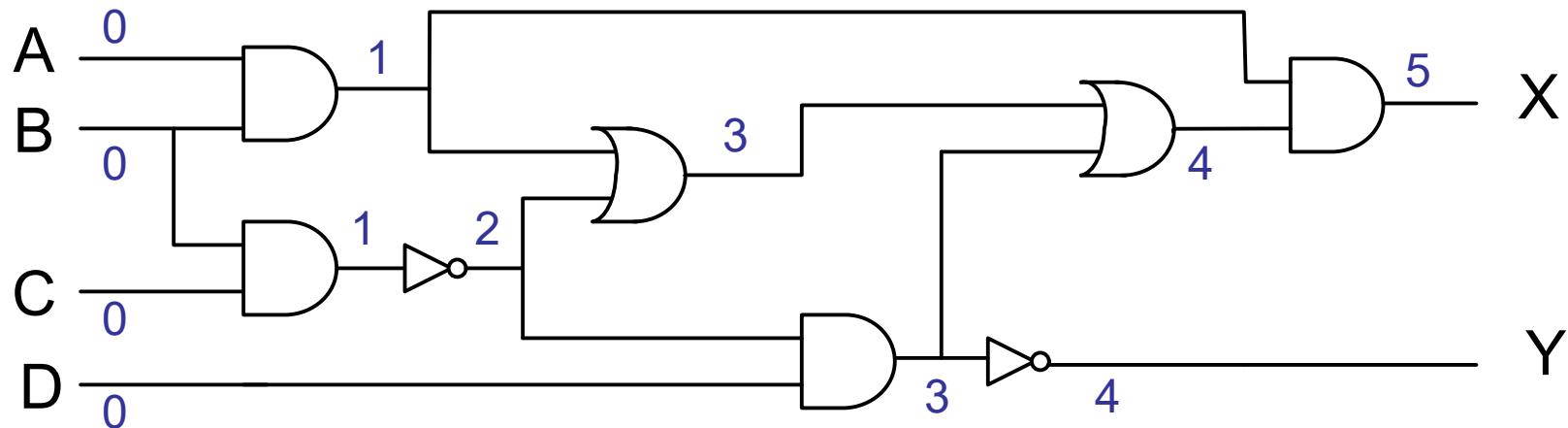
Path Delay

Path Delay: Time to propagate through a series of combinational gates.

- Note:
 - Gate output changes only after the last input arrives
 - Use the longest path / last-arriving input time for each gate

Example:

- Assume each gate delay is the same (1ns in this example)

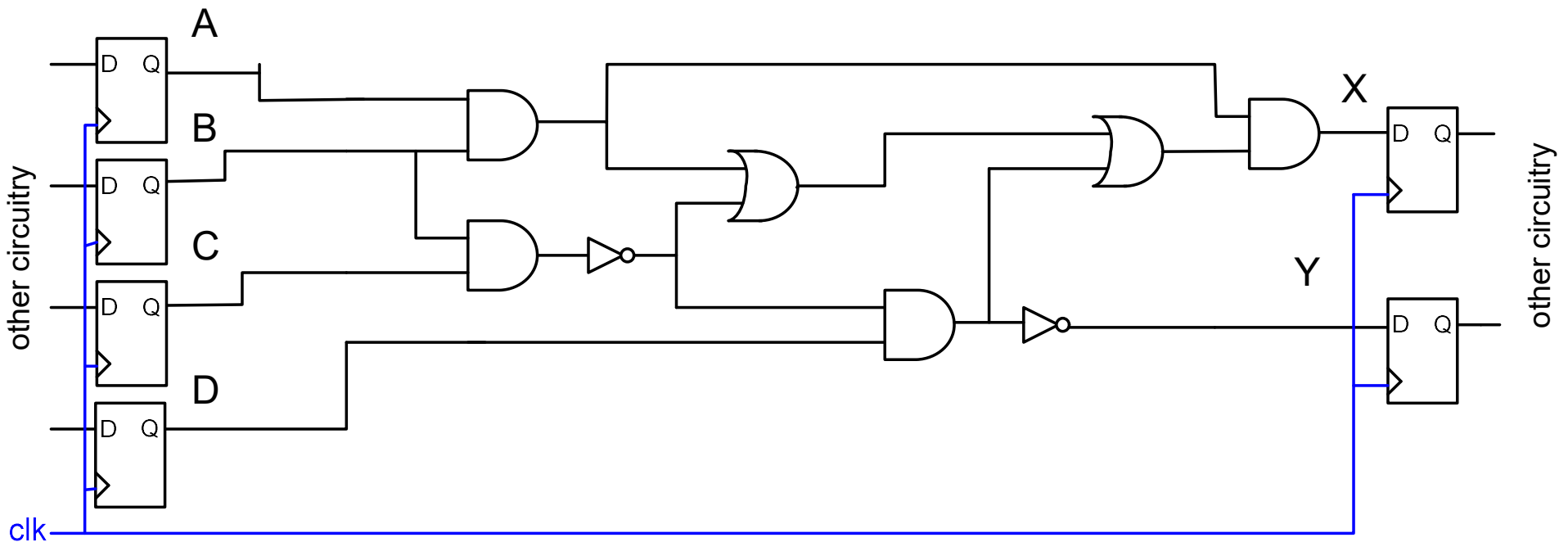


In other words, in the previous circuit, if we apply all inputs at time 0, then after 5ns, all the outputs have settled to their final values.

- Notes:
1. Some outputs might settle earlier
 2. Some outputs may switch back and forth a few times before settling to a final value

How to synchronize inputs?

Add flip-flops!



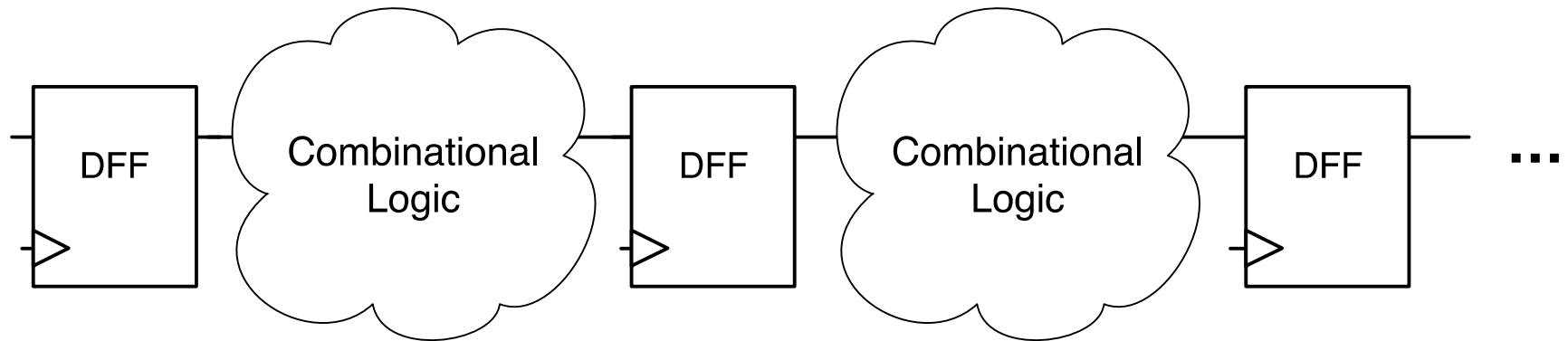
Rising edge on clock at time 0. Assuming no delay in the flip-flops, the outputs of the source (left four) flip-flops change at time 0.

Some time later (one clock cycle), the clock goes high again, and the destination flip-flops read in X and Y

How fast can we run the clock?

How fast can we run the clock?

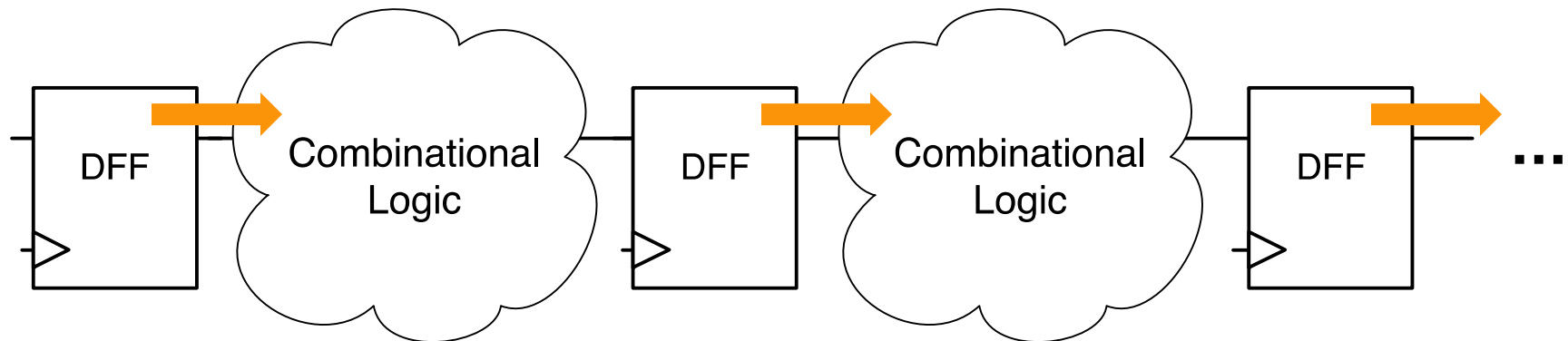
Clock's purpose is to tell flip-flops when to read in data



How fast can we run the clock?

Key Idea:

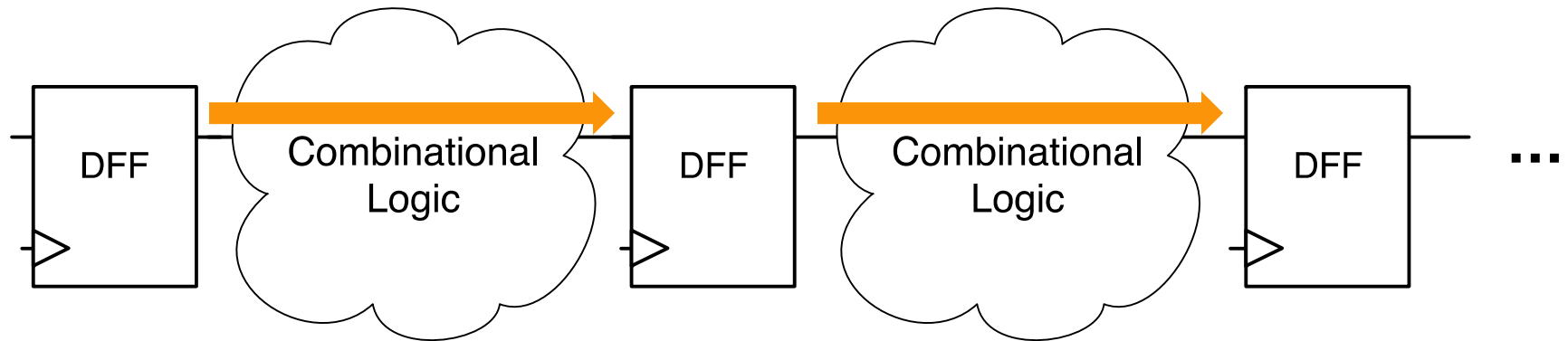
1. On clock edge, new inputs sent into combinational logic



How fast can we run the clock?

Key Idea:

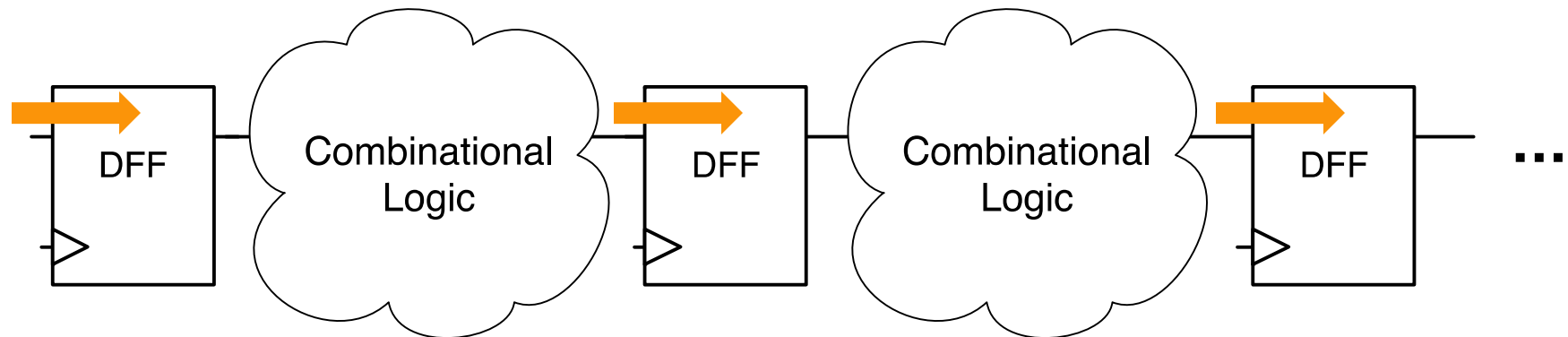
1. On clock edge, new inputs sent into combinational logic
2. Takes some time for outputs to settle



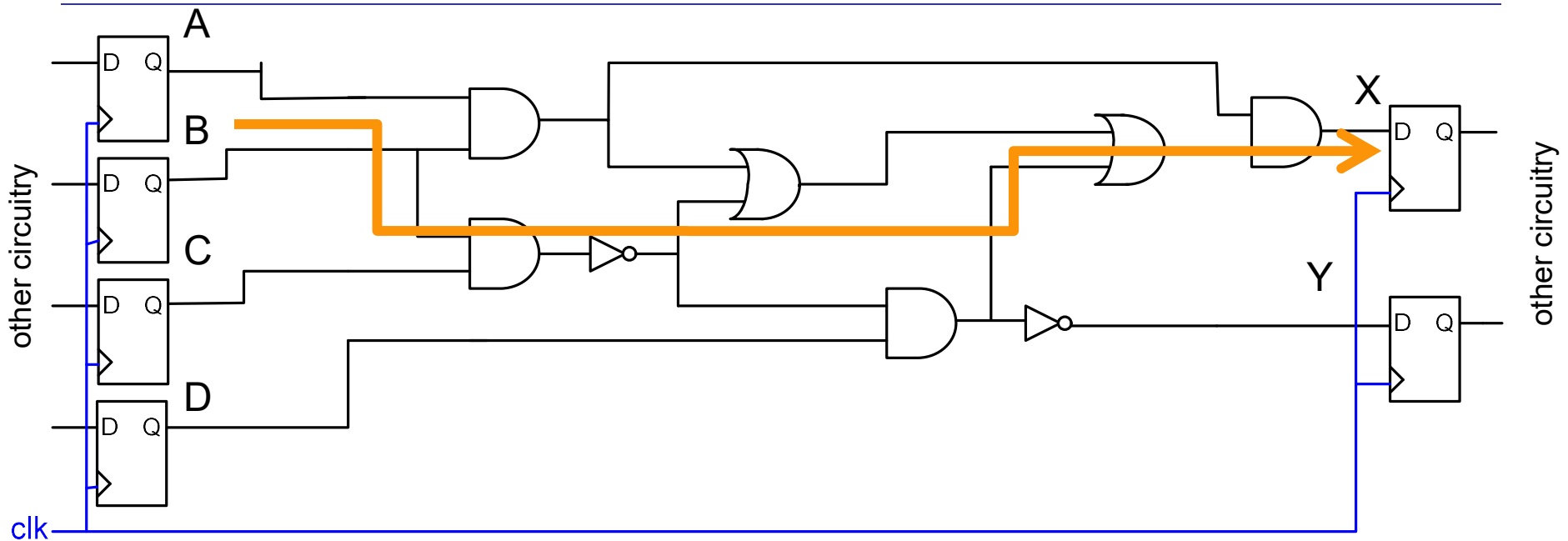
How fast can we run the clock?

Key Idea:

1. On clock edge, new inputs sent into combinational logic
2. Takes some time for outputs to settle
3. Don't want to read in new outputs before they are ready



Example: Critical Path Delay



Critical Path is

B → AND → OR → OR → AND → X

Critical Path Delay is 5ns

Clock period cannot be smaller than 5ns or else X register will read in wrong data

Max Clock Frequency / Min Clock Period

Critical path delay limits the minimum clock period
(hence the maximum clock frequency)

$$t_{ClockMin} \geq t_{CriticalPath}$$

From the example

$$t_{CriticalPath} = 5ns$$

$$\rightarrow t_{ClockMin} = 5ns$$

$$\rightarrow f_{ClockMax} = \frac{1}{5ns} = 200MHz$$

What happens if you run the clock slower?

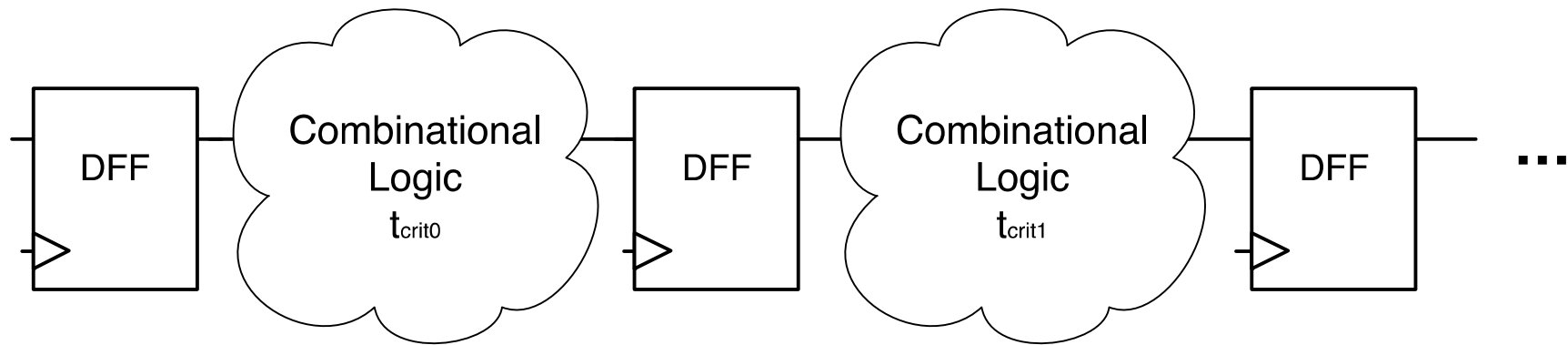
$$f_{Clock} < f_{ClockMax} \quad \Leftrightarrow \quad t_{Clock} > t_{CriticalPath}$$

No problem.

Combinational logic has more time to settle

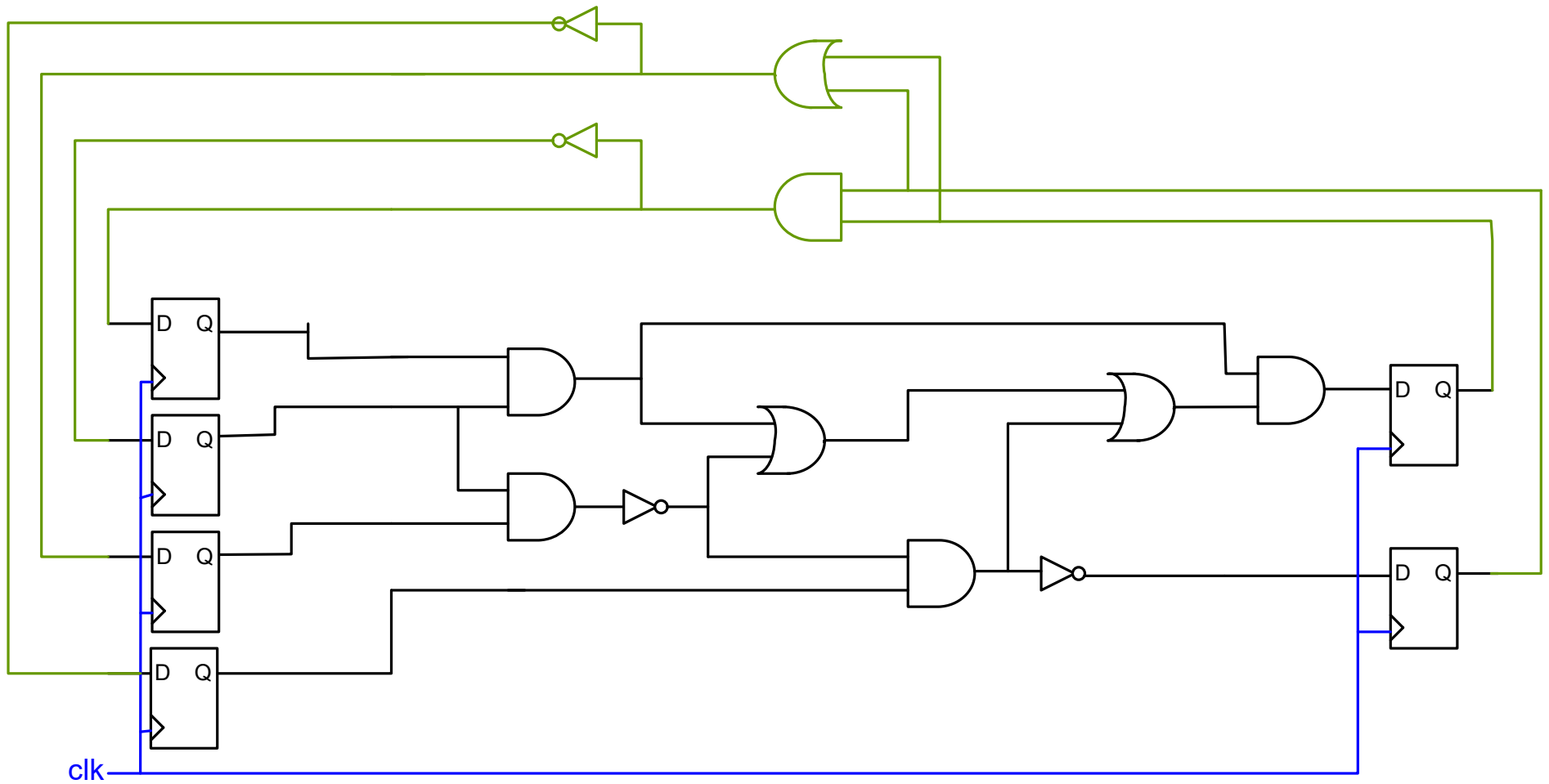
Critical Path in Entire Design

Since a **single clock** controls all flip-flops in all stages, you need to look for the critical path amongst **all stages**



$$t_{CriticalPath} = \max_{foreach\ path\ i} (t_{crit_i})$$

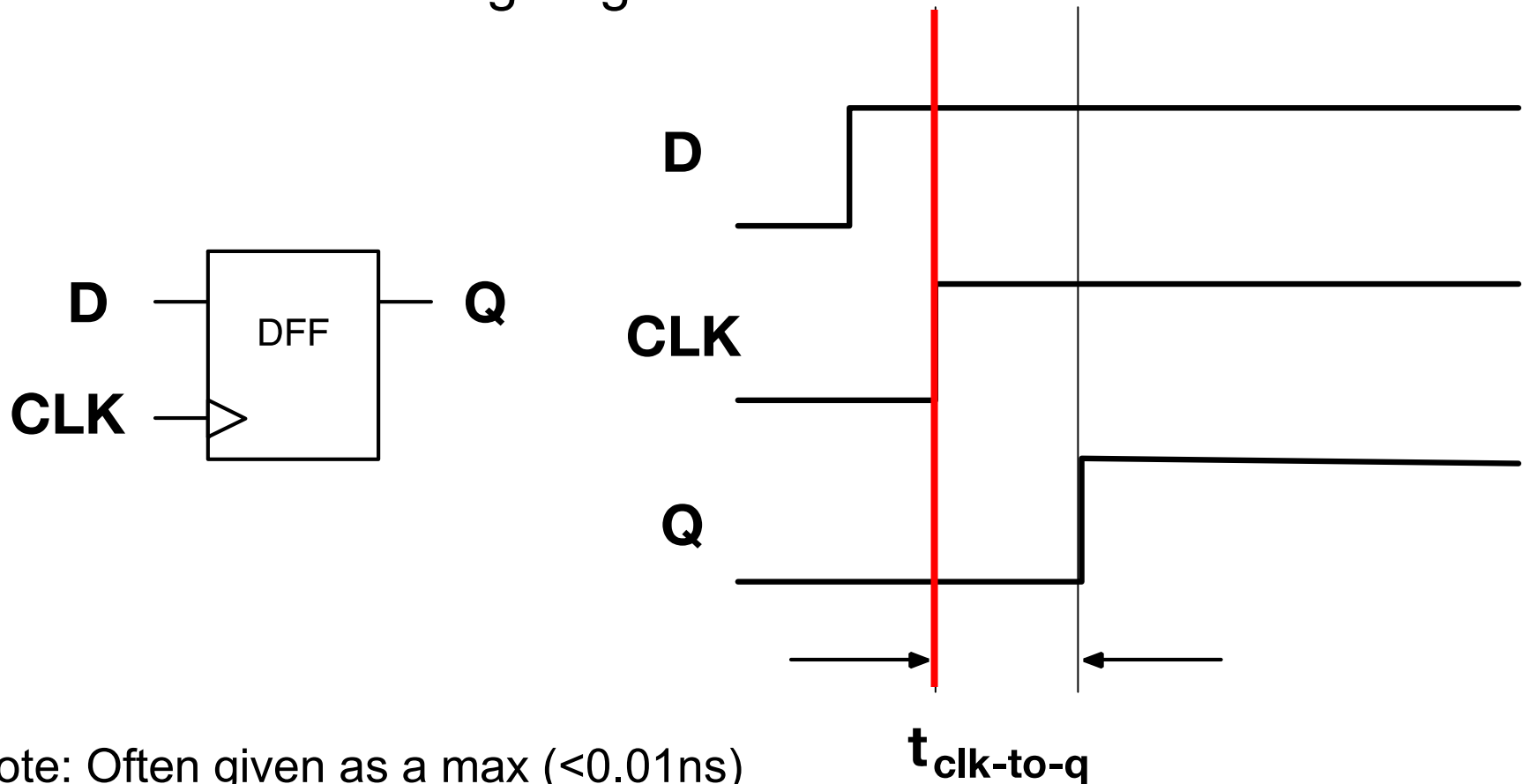
For Example



Flip-Flop Timing Constraints

Clock-to-Q Delay

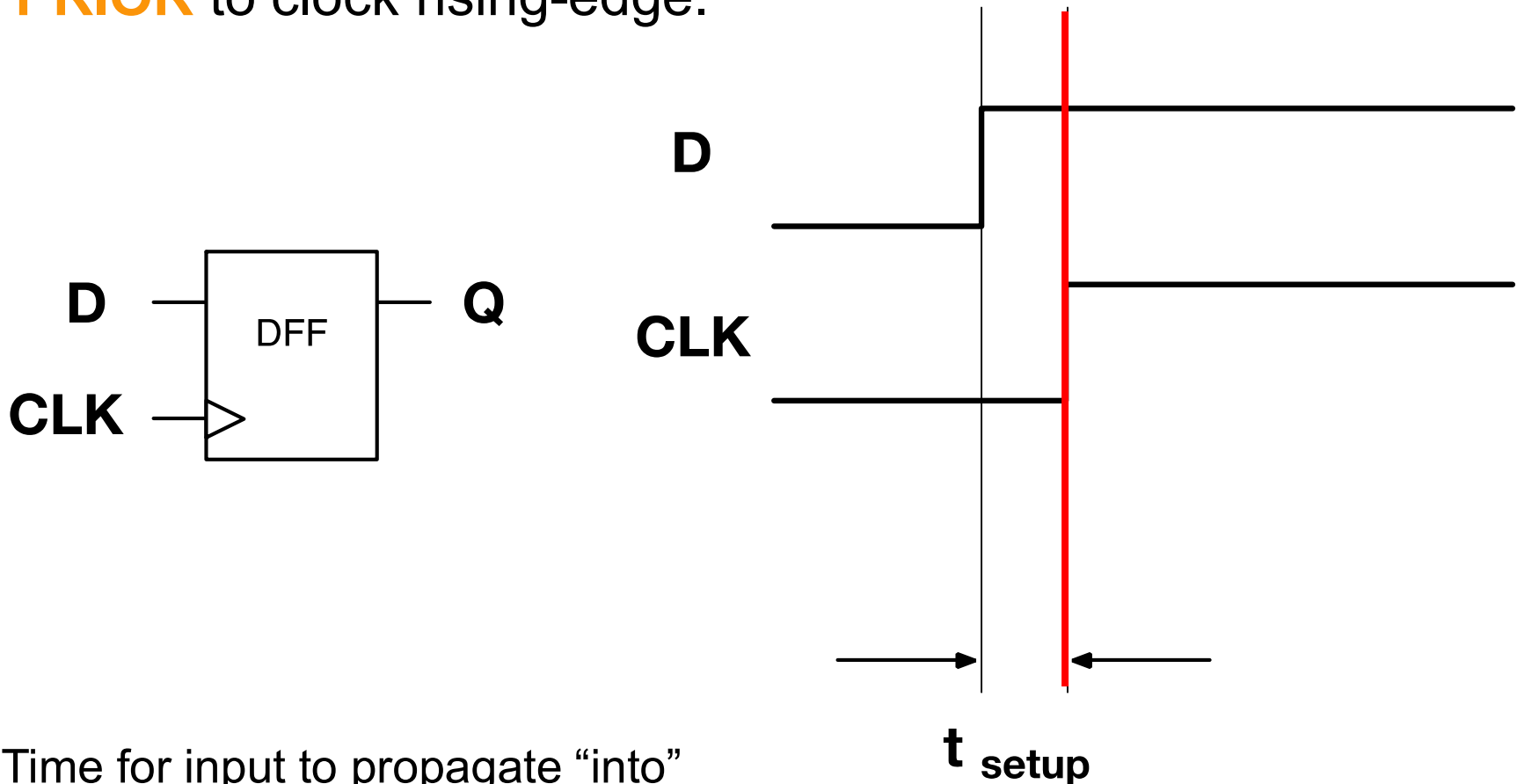
The time that it takes for the output of a Flip-Flop to settle after the clock rising-edge occurs



Note: Often given as a max (<0.01ns)
and a min value (>0.001ns).
Always very fast!

Setup Time

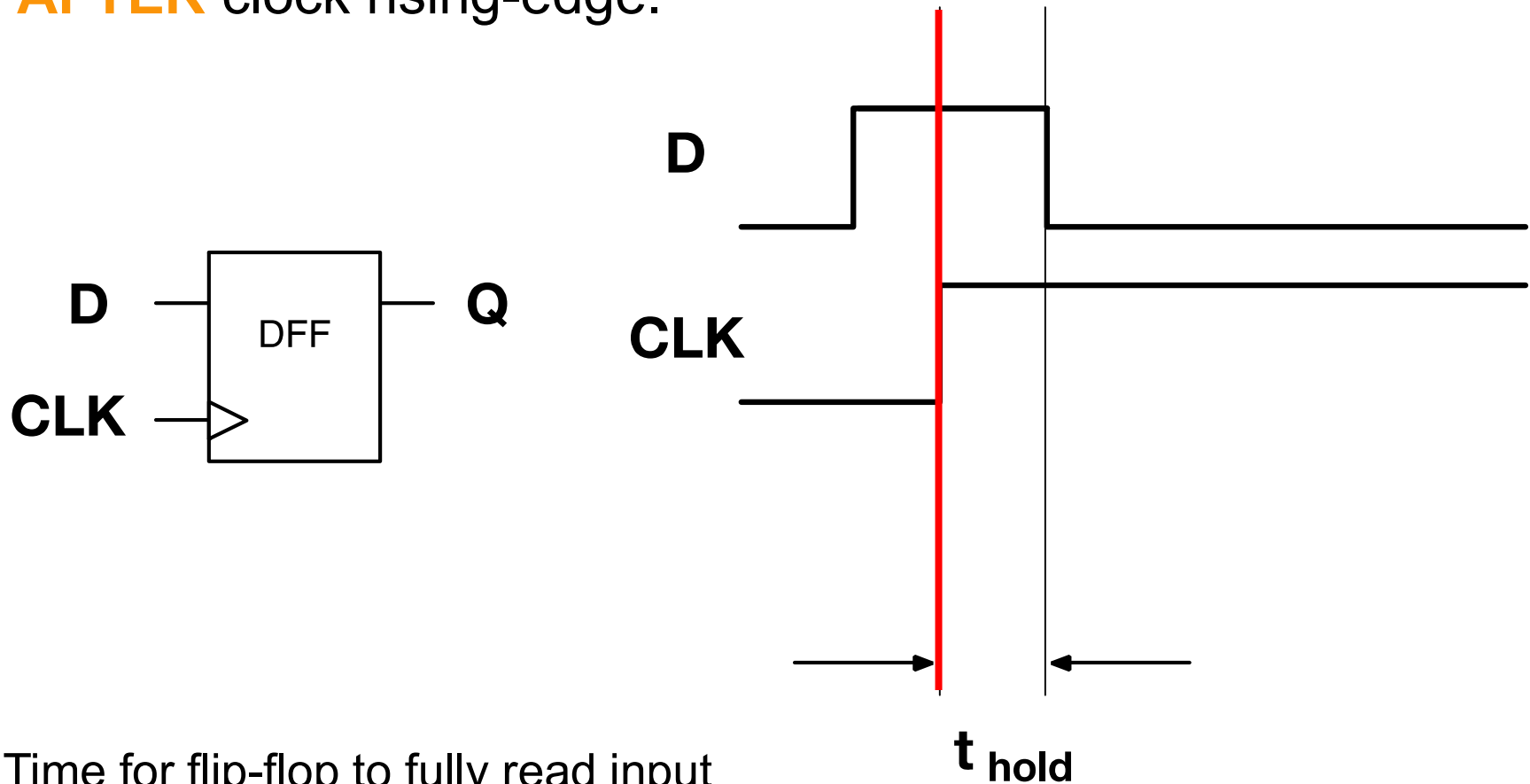
Input to flip-flop cannot change for a certain amount of time **PRIOR** to clock rising-edge.



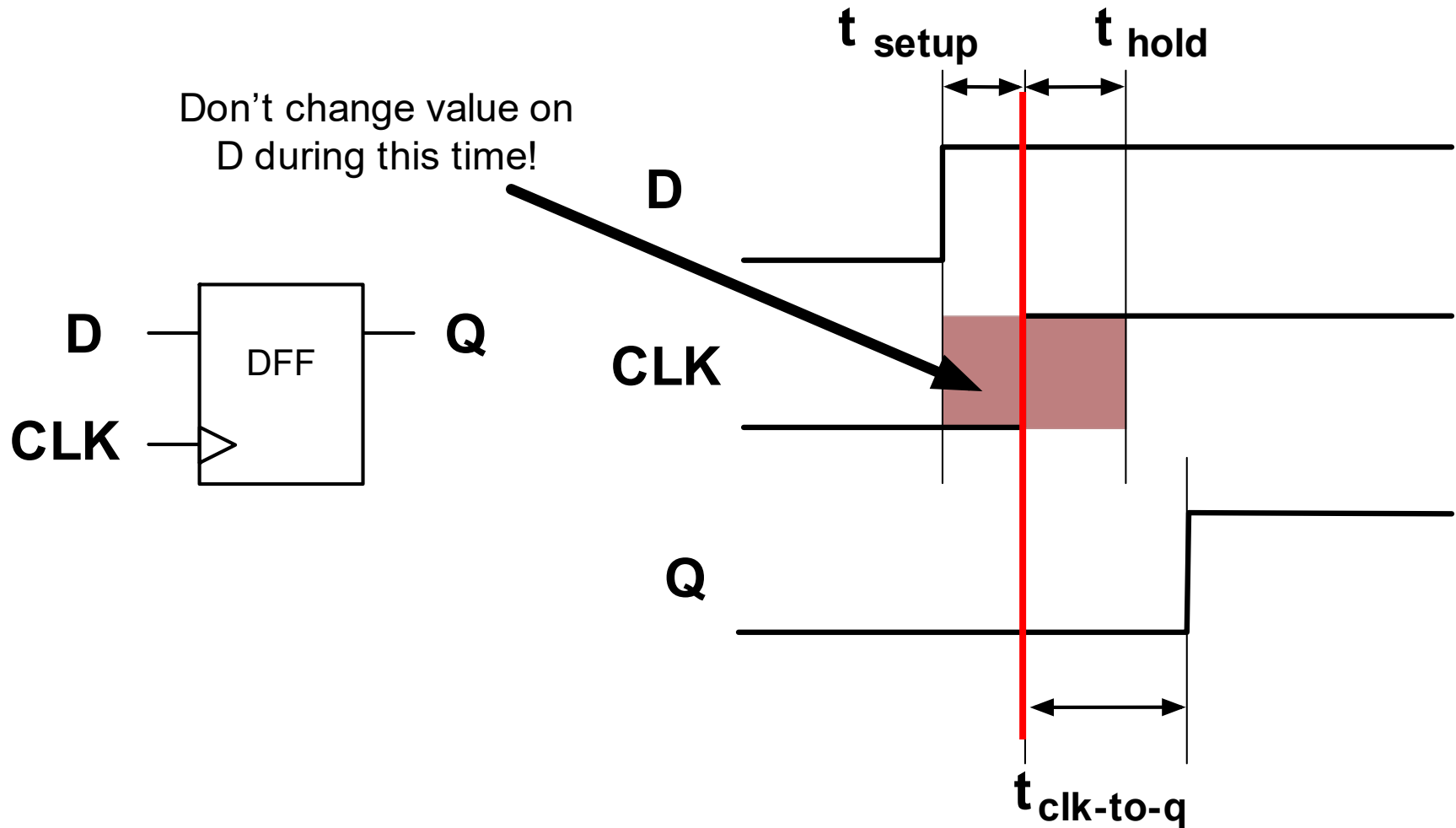
Time for input to propagate “into”
the flip-flop before clock arrives

Hold Time

Input to flip-flop cannot change for a certain amount of time **AFTER** clock rising-edge.



Summary of Flip-Flop Timing



Two Requirements

Setup Requirement

Input values must be ready for destination flop ahead of when the rising clock edge arrives.

Hold Requirement

Input values of destination flop must not change for a short time after rising clock edge.

How fast can we run the clock?

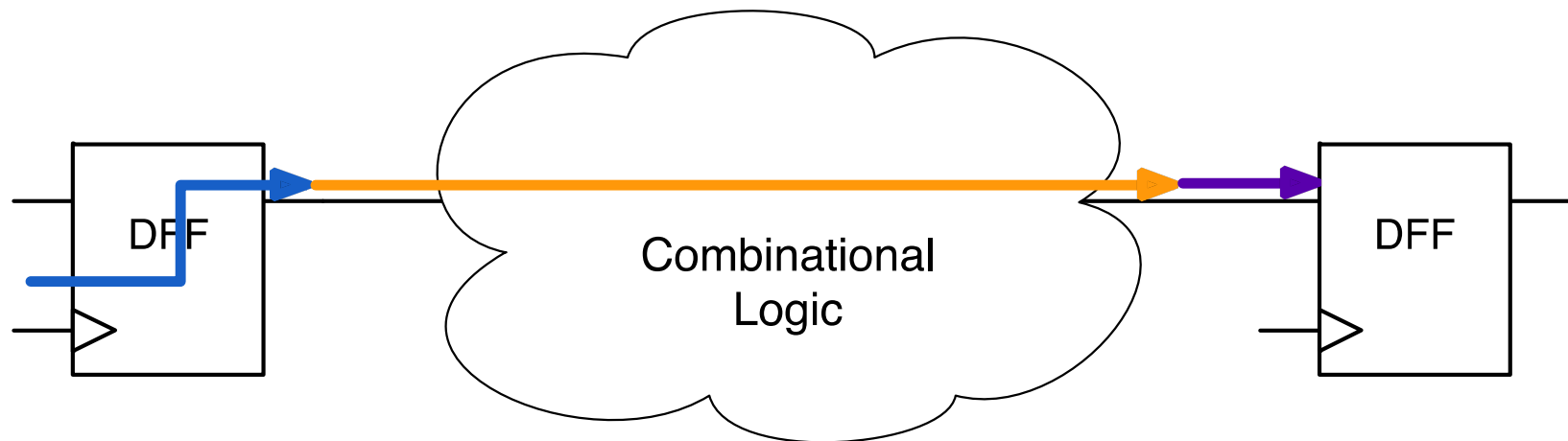
Revisited to account for flip-flop constraints

Setup Requirement

$$t_{Clock} \geq t_{clk-to-q\max} + t_{CriticalPath} + t_{setup}$$

After clock rising-edge, there must be enough time so that

1. source flip-flop's output can settle to correct value
2. critical path can settle to correct value
3. value arrives at destination flip-flop early enough for setup time



Key Concept for Setup Time:

**Must have enough time for
input of FF to settle
before FF reads in value**

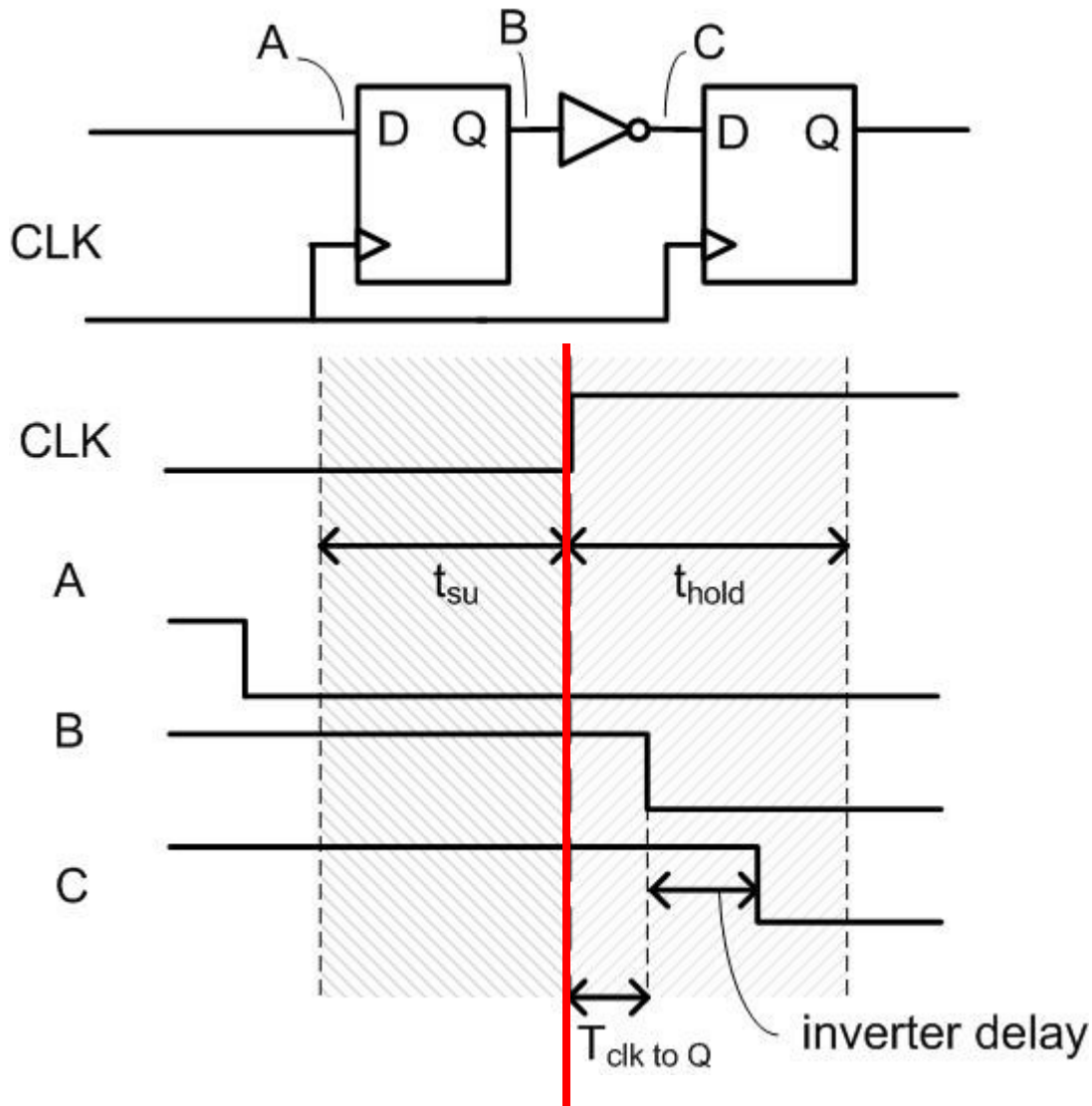
Hold Requirement

Remember, at the rising edge of clock, FF is now reading in data.

Input to the FF cannot change for t_{hold} amount of time

- Unlikely to be a problem for Critical Path
- Need to be careful for the **fast (non-critical) paths**

Suppose the hold time is really big...



@posedge clock:

FF A changes (t_{setup} ok)

FF C changes (t_{setup} ok)

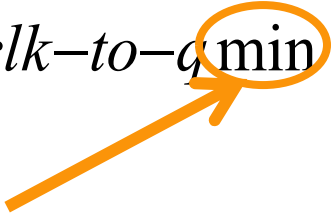
FF A hold time is ok

FF C hold time is violated
due to fast path A→B→C

NOTE: increasing clock
period doesn't fix this,
because violation is caused
by a race from the same
clock edge, not two edges.

Hold Requirement

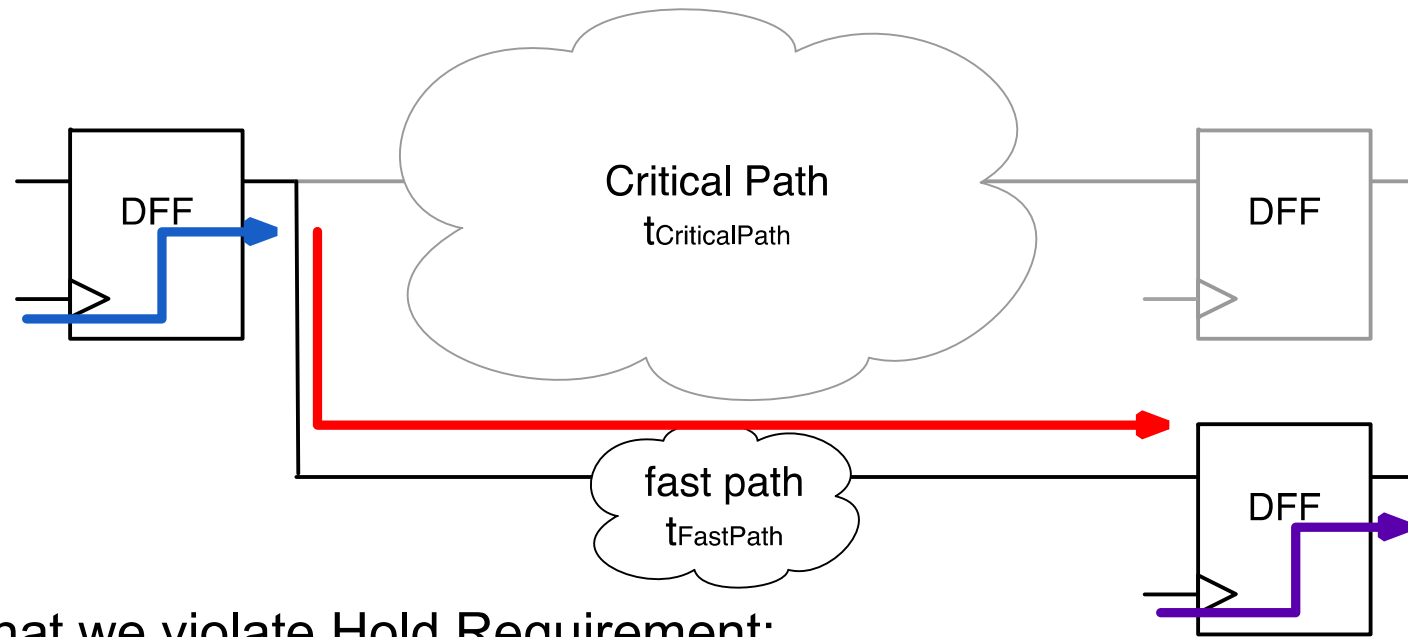
In general, for all paths:

$$t_{clk-to-q\ min} + t_{path} \geq t_{hold}$$


Minimum clk-to-q delay because we need to design for the fastest possible arrival time of signal to destination flop

Note: On each clock cycle, flip-flop is BOTH reading in a new value (when you look at it as a destination flip-flop) AND providing a value to the next stage (when you look at it as a source flip-flop)

Hold Violation



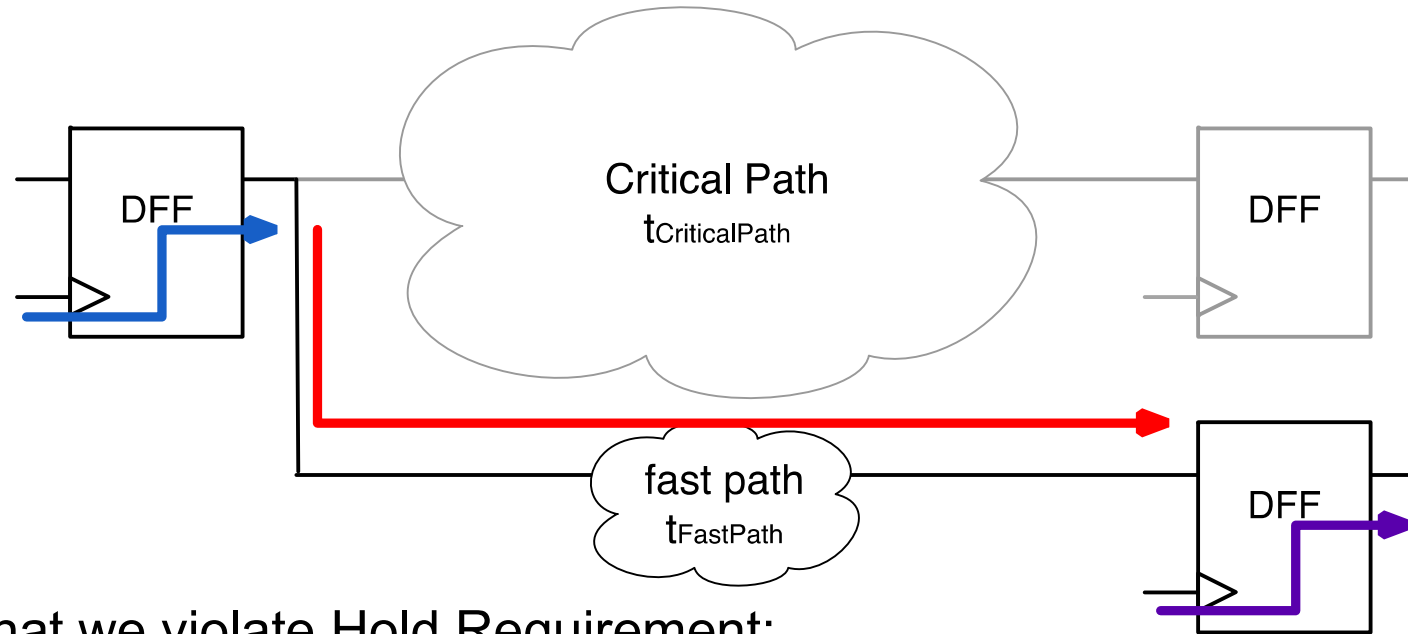
Say that we violate Hold Requirement:

$$t_{clk-to-q\min} + t_{FastPath} < t_{hold}$$

After clock rising-edge,

1. source flip-flop's output settles
2. fast path settles SUPER QUICKLY to NEW value
3. destination flop still wants to see OLD value, but NEW value is present (i.e. hold time has not yet passed for that reading) → hold time violation

Hold Violation



Say that we violate Hold Requirement:

$$t_{clk-to-q\ min} + t_{FastPath} < t_{hold}$$

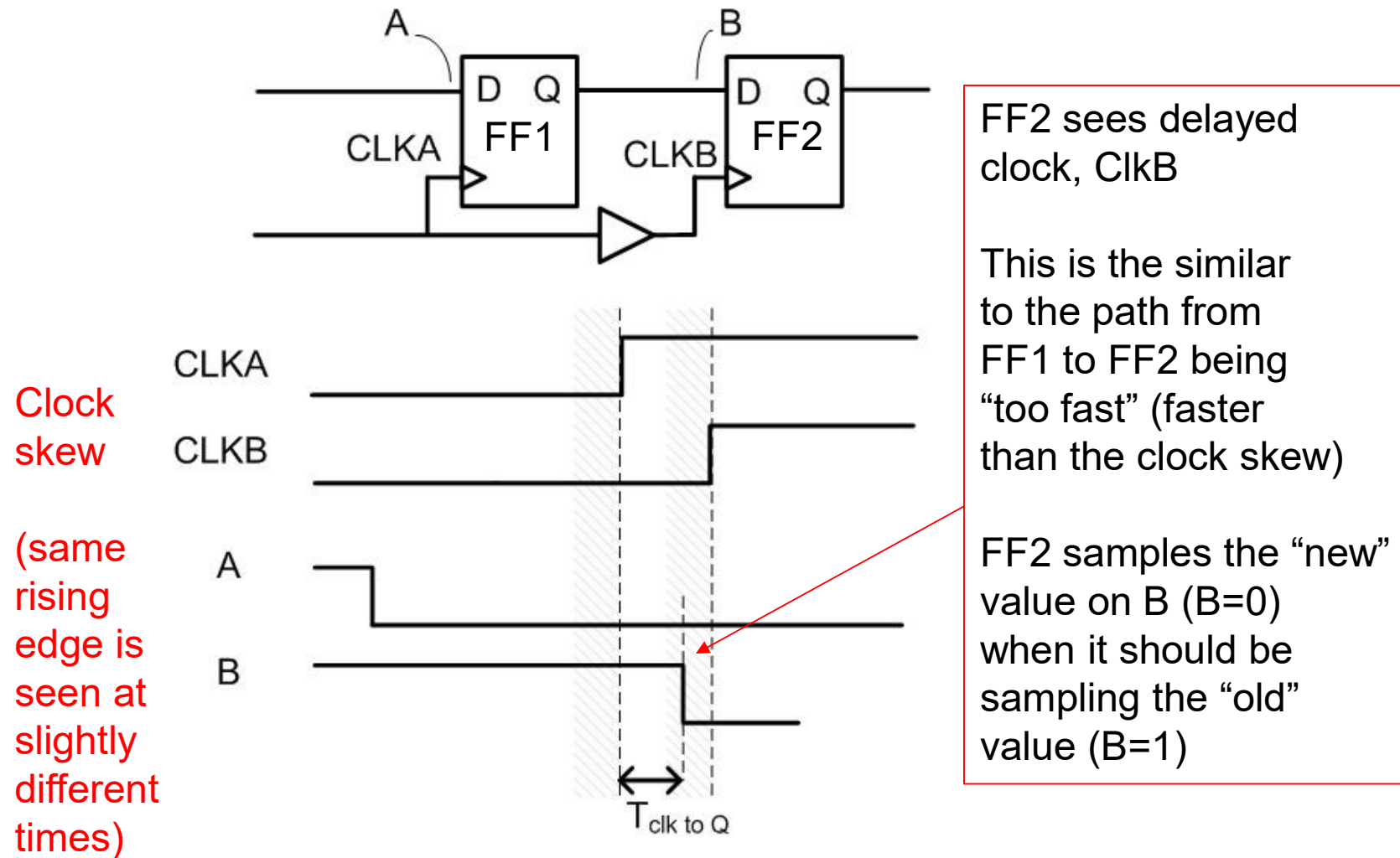
Note:

t_{Clock}
isn't in the equation

New data arrives via fast path on same clock edge and confuses this FF.

Since the problem occurs on the same rising clock edge on both FFs, slowing the clock doesn't fix it (slowing the clock only changes the distance between two clock edges).

This is often a problem if the clock is delayed
(due to long wires, for example)



We will come back to clock skew later

How To Fix Violations

Setup Violation

- Slow down clock
- Move registers around (reduce length of critical path)

Hold Violation

- **CANNOT** address by changing clock frequency!
- Lengthen path by adding gates or interconnect wire
- If possible, reduce clock delays and clock skew
(try not to add clock delays or increase clock skew)

**GOOD NEWS! If you are using an FPGA,
the CAD tools handle all this for you!**
(sort of, not really)

Learning Objectives

1. Understand the RC model for gate delay and where it comes from
2. Be able to find the critical path, and hence the maximum clock speed, of a logic circuit (with or without flip-flops)
3. To understand set-up time, clk-q time, and hold time of a flip-flop
4. Understand setup and hold timing requirements of a design
5. Given a circuit, be able to calculate the maximum clock frequency
6. Given a circuit, be able to calculate the minimum hold time on the flip-flops